10-2018

Exploring experiential learning model and risk management process for an undergraduate software architecture course

Eng Lieh OUH
Singapore Management University, elouh@smu.edu.sg

Yunghans IRAWAN

Exploring Experiential Learning Model and Risk Management Process for an Undergraduate Software Architecture Course

Ouh Eng Lieh
School of Information Systems
Singapore Management University
elouh@smu.edu.sg

Yunghans Irawan
Institute of Systems Science
National University of Singapore
yirawan@nus.edu.sg

Abstract— **This paper shares our insights on exploring the experiential learning model and risk management process to design an undergraduate software architecture course. The key challenge for undergraduate students to appreciate software architecture design is usually their limited experience in the software industry. In software architecture, the high-level design principles are heuristics lacking the absoluteness of first principles which for inexperienced undergraduate students, this is a frustrating divergence from what they used to value. From an educator's perspective, teaching software architecture requires contending with the problem of how to express this level of abstraction practically and also make the learning realistic. In this paper, we propose a model adapting the concepts of experiential learning and risk management to design the course on software architecture. The lesson plan promotes active learning with activities to observe how different parts of an information systems architecture work, experience the impact of real software quality issues or risks arise, reflect on the root causes of these risks, conceptualize and subsequently implement the countermeasure to mitigate the risk. We divide the course into first session conducted based on the traditional lecture format and second session based on our proposed experiential risk learning model. We evaluate the feedback ratings of 128 undergraduate students of an information system program for the two sessions and perform sentiment analysis on their comments. We also generalize the applicability of our experiential risk learning model to courses in other domains of software engineering. The key contribution of this paper is the experiential risk learning model. We hope that this model alleviates the challenge to design a software architecture course for undergraduates and can be used as another teaching method for active learning.**

Keywords—software architecture, experiential learning, risk management process, active learning

I. INTRODUCTION

Software architecture given its level of abstraction, remains a difficult subject for learners to grasp and for educators to teach. The skill set possessed by a software architect is also multi-faceted which increase the level of difficulty for one be a competent software architect.

The role of a software architect typically entails one to have t*echnical skills* that minimally include software design and programming experiences. Though software architects do not need to be technology experts, it is essential that the architect keeps abreast of the frequently changing technology trends. *Analytical skills* are also essential for the software architect to grasp the problem quickly, diagnose the possible root causes and make significant decisions for the project. Software architecture is the fundamental organization of a system embodied in its *components*, their *relationships* to each other, and to the *environment*, and the *principles* guiding its design and evolution [1]. An architect who is unable to make significant design decisions (*principles*) on the *components* and their *relationships* in an *environment* where much is unknown, where there is insufficient time to explore all alternatives, and where there is pressure to deliver is unlikely to succeed. The life of a software architect is a long and rapid succession of suboptimal design decisions taken partly in the dark [2]. These significant architecture design decisions can affect the service variability and ultimately the service profitability of the system [3]. Lastly, the architect should have effective *communication skills* to understand and negotiate project requirements with relevant stakeholders. The development of these skills usually comes with the experiences of an individual having the opportunities to solve or at least observe architectural problems in an industry project.

The key challenge to design a software architecture course for undergraduate students is how to teach abstract software architecture concepts to students who, usually, have very limited experience in the software industry. A systematic problem in getting architecture across, to students lacking real-world design experience, is this – the high-level design principles are heuristics lacking the absoluteness of first principles. All notions agree that designing software architecture is about linking various related concepts to achieve quality concerns of software. This is different to other areas of software engineering. For example, we have a clear definition of what software testing and maintenance are and why they are essential. However, how one defines architecture ultimately depends on the context, the stakeholders, the concerns, and eventually on what the purpose of the architecture.

In this paper, we attempt to address the above key challenge by proposing a model adapting the experiential learning model and risk management process. We termed this model as experiential risk learning model. The teaching method based on this model comprises of activities to simulate risks that can happen in practical scenarios and their role is to be able to recognize these risks, reflect on the causes and

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

mitigate these risks. We implement a conducive environment where the student can experience the risk identification, analysis and resolution for a software architecture. During this process, the students not only get to learn the technicality of a software architecture design, but they will also get to analyze the risk symptoms and practice their skills to effectively communicate the resolutions of these risks, all of which are the essential skills required of an architect.

We extend the applicability of our proposed model to the software engineering domain in general. The key contribution of this paper is our proposed experiential risk learning model. The rest of the paper is organized as follows: We first present related work in Section II. We describe the background information for our study in Section III. Our proposed experiential risk learning model is discussed in Section IV. We give the course structure and goals in Section V and describe how we apply the proposed model in our course in Section VI. Validations of our results are described in Section VII and threats to the validity of our result are discussed in Section VIII. We extend the applicability of our proposed model to the software engineering domain in Section IX and our conclusion is in Section X.

II. Related Work

Challenges in teaching software architecture courses to undergraduates: Rupakheti and Chenoweth in [4] described their experiences and learnings in teaching software architecture course to undergraduates. The systematic problem in getting architecture across to undergraduates especially for learners primarily with programming skills lacking real-world design experience. While the overall design of the software architecture course at the undergraduate level consider how ambitious the course designer's goals are given the students are unlikely to take the role of the architect in their first job. Galster and Angelov [5] describe the framework involving the relationship of concept (software architecture), representation (architecture description), referent (software architecture practice) to the learner element in the learning space. In addition to the vagueness of the concept of software architecture itself, architecture problems are usually "wicked". Asking students to create an architecture is different to e.g., asking them to write a Java program - students have a much clearer understanding of what the expected outcome is.

People learn best through experience: Kolb [6] offers a systematic statement of the theory of experiential learning and its modern applications to education, work, and adult development. Kolb models the underlying structures of the learning process based on the latest insights in psychology, philosophy, and physiology. Building on his comprehensive structural model, he offers an exceptionally useful typology of individual learning styles and corresponding structures of knowledge in different academic disciplines and careers. Appropriate assessment methods that can be aligned with the learning outcomes of experiential learning. Mitchell and Delaney [7] describe four core issues that need to be addressed when assessing students in a group project setting and these are assessment of group performance, individual contribution, project deliverables and the course success.

Principles and practices in software risk management: Boehm [8] describes the discipline of software risk management to formalize the risk-oriented correlates of success into a readily applicable set of principles and practices. His survey shows that most postmortems of the software project disasters have indicated that their problems would have been avoided or strongly reduced if there had been an explicit early concern with identifying and resolving their high-risk elements. Experiences in the technology and management environment play a part in the probability of the risk and impact for cost failure. To handle all the complex people-oriented and technology-driven success factors in projects, a significant measure of human judgment is required.

Risk plays a pivotal role in experiential education: Lidde [9] describes the essential role of the risk element in the learning process. Without unknown outcomes and the motivation to explore them, experiential education probably would not have existed. Experiential educators have an obligation to create opportunities for clients where they face the unknown and preserve despite the potential for significant loss.

III. Background

A. Experiential Learning Model

Experiential learning is the process whereby knowledge is created through the transformation of experience. Kolb [5] theory of experiential learning presents a cyclical model of learning through four stages. The first stage of *Concrete Experience (CE)* requires the learner to actively experience an activity. In the second stage of *Reflective Observation (RO),* the learner consciously reflects on their experiences. For the third stage of *Abstract Conceptualization (AC),* the learner attempts to conceptualize a theory or model of what is observed. The last stage of *Active Experimentation (AE)* is where the learner plans how to test a model or theory.

B. Risk Management Process

Risk is defined as the likelihood of a future event having a negative consequence. If a threat exploits a vulnerability that results in an impact to a valuable asset, there is a risk that needs to be mitigated with controls or countermeasures. Risk management process is therefore the identification, evaluation, and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor, and control the probability or impact of unfortunate events.

The specific steps of a software risk management are *risk identification, risk analysis, risk prioritization, risk-management planning, risk resolution and risk monitoring* [8]. Risk identification produces lists of the project-specific risk items likely to compromise a project's success. Risk analysis assesses the loss probability and loss magnitude for each identified risk item, and it assesses compound risks in risk-item interactions. Risk prioritization produces a ranked ordering of the risk items identified and analyzed. Risk-management planning helps prepare one to address each risk item (for example, via information buying, risk avoidance, risk transfer, or risk reduction), including the coordination of the individual

risk-item plans with each other and with the overall project plan. Risk resolution produces a situation in which the risk items are eliminated or otherwise resolved. Risk monitoring involves tracking the project's progress toward resolving its risk items and taking corrective action where appropriate. Residual risks are the remaining risks after application of the controls. Acceptance of these residual risks by the system owner is required before the system can be deployed and launched.

We map how the steps in the risk management process can be related and practiced within the stages of the experiential learning model in Table I. We describe the proposed experiential risk learning model based on this relationship in the next section and evaluate its effectiveness with the course feedbacks in Section VII.

IV. Proposed Experiential Risk Learning Model for Software Architecture Course

Fig.1 shows our proposed experiential risk learning model and how the experiential learning model based on Kolb and risk management process can be applied to the design of a software architecture course. We adopted the same four stages in Kolb model and inject the steps of the risk management process within each stage.

The concrete experience stage comprises of the learner observes risk scenarios whereby independent events can happen to an actual system, resulting in a non-trivial negative impact. These scenarios should be repeatable to ensure that it does not happen by chance. The input to this stage is an initial risk scenario log which is empty in the first cycle. The output of this stage is the updated risk scenario log.

The reflective observation stage involves the learner attempting to evaluate the severity of these risks. During this stage, the learner can have more guidance from the instructor or additional lessons on underpinning knowledge leading to this compromise. These underpinning knowledge are the essential technical or non-technical information that is related to the event generated, how this event can access the system or how the system works. With this knowledge, the learner is expected to prioritize the items in the risk scenario log and determine the key risks to be addressed in this cycle. The input to this stage is the risk scenario log and the output from this stage is the prioritized risks.

During the abstract conceptualization stage, the learner needs to model a solution to mitigate the prioritized risks. The solution model must be able to effectively mitigate the same risk from happening and should be clearly defined, implementable and testable. The solution model might introduce more risk scenarios and these can be added to the risk scenario log but not required to be addressed in the same cycle. The input to this stage is the prioritized risks and the output is the solution model and an updated risk scenario log.

The active experimentation stage involves the learner to implement the solution and execute it to mitigate the prioritized risks. It is again noted that there can be additional risk scenarios resulting from the concrete solution but the effect of this solution should be permanent for the prioritized risks in this cycle. The input to this stage is the solution model and the output is the concrete solution and updated risk scenario log.

The cycle repeats with the learner entering the concrete experience stage again but with the updated risk scenario log from the earlier cycle and the learner can attempt to address other risks in another cycle. For our course, this model is applied to evaluate multiple software qualities of a given architecture. There is a side effect to this -the implementation of a solution to mitigate the risks of one software quality might trade-off another software quality. For example, the implementation of secure socket layer (SSL) to mitigate against sniffing will likely increase the time required to process the web request, negatively impact the performance of the system. Another example is the development of a product-line improves maintainability but tradeoff additional upfront costs as described in industrial contexts by Khue, Ouh and Stan for a set of related mobile apps [10] and by Koznov, Luciv, Basit, Ouh and Smirnov [11] for a set of related software technical documentation. The students have to be aware of and take these factors into considerations.

V. Course Background

A. Undergraduate Students

The participants in this study comprised of 128 students in their third and fourth year of an Information Systems program taking a software architecture course. These students attended the course in four different classes at different time slots each week and each class comprising of 28-36 students. The same instructors conduct the class for all the four classes. They have successfully completed a series of compulsory courses in programming, design and project management before embarking on this architectural analysis course. In this paper, we collectively refer to them as learners.

B. Course Goals, Structure and Assessment

The primary goal of the course is to prepare learners to appreciate software architecture designs as they are unlikely to be designing complex software architectures at the start of their IT career. We refer to the course in this study as architectural analysis (AA).

The AA course is conducted within the semester weeks with a 3 hours session per week. There are two key sessions that we evaluated for this study. Session 1 comprises of topics in software architecture fundamentals and accounts for 30% of the total duration for the two sessions combined. Session 2 comprises of topics to design for software qualities and accounts for the remaining 70% of the duration. The longer duration for session 2 is due to the longer time taken to conduct the practical hands-on activities. The teaching methods applied in session 1 are mainly lecture-style with slides, class exercises and quizzes while the proposed experiential risk learning model is applied in session 2. The details of each session are further described in Section VI.

The assessments for this course is based on class participation of the students and their graded deliverables in both session 1 and 2, followed by a written examination at the end of the semester. The assessments for session 1 involves in-

TABLE I. RELATIONSHIP BETWEEN THE EXPERIENTIAL LEARNING MODEL AND RISK MANAGEMENT PROCESS.

Experiential Learning Model	Risk Management Process
Concrete Experience (CE)	Risk Identification
Reflective Observation (RO)	Risk Analysis Risk Prioritization
Abstract Conceptualization (AC)	Risk Management Planning
Active Experimentation (AE)	Risk Resolution Risk Monitoring

class quizzes and exercises conducted after the slide lectures. The assessments for session 2 involves activities for each stage of the proposed model and they have to address the risk problems of one stage before proceeding to the next. There are many software qualities in the design of a software architecture and the students get the opportunities to apply the proposed model for a number of the software qualities we designed for them. At the end of session 2, the students are required to present their risk findings for all the software qualities. For the final written examination, we extend the scenarios discussed in class exercises and risks encountered in session 1 and 2 respectively to evaluate their understanding and ability to apply the same concepts in a different context.

VI. COURSE DESIGN

A. *Software Architecture Fundamentals*

This session 1 on software architecture fundamentals exposes the students to the essential concepts to communicate and analyze software architectures. The students practice communication of software architecture designs with class exercises using sequence, deployment and network diagrams. We also introduce the basic concepts in operating system and systems networking as a common baseline for the students as some of them are not being exposed in earlier modules. This session provides the essential underpinning knowledge for them to apply in the next session.

B. *Analysing Designs for Software Qualities*

In this session 2, we focus on three essential software qualities – Availability, Security and Performance based on ISO 25010 [12]. For each of the quality, we apply the four stages of experiential learning for the learners to experience the qualities in action, reflect, conceptualize and experiment for further insights. The design of this session follows a similar format where the students will form groups, experience the situations where the quality is being compromised. With the observations, they reflect on the root causes leading to the system being compromised and the impact to the system. They subsequently proceed to decide what they can do differently to improve the quality and mitigate the risks. The fourth stage allows them to experiment with their ideas and evaluate whether it can mitigate the risks identified. This cycle of experiential learning for each quality iterates and after one

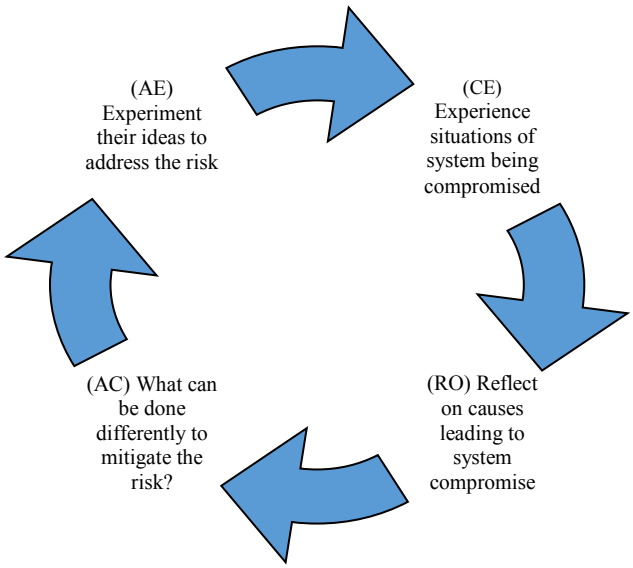

FIG. 1. PROPOSED EXPERIENTIAL RISK LEARNING MODEL

cycle, additional potential new risks can be introduced. One cycle of the concrete experience, reflective observation and abstract conceptualization stages are conducted within the lesson time but the active experimentation and subsequent cycles can be carried out within and outside lesson time.

The initial configuration involves the students setting up the given java proxy server and a Tomcat web application hosting on two devices connected over a switched local area network. This setup is a two-tier architecture with the browser client accessing the proxy server which in turn access the web application.

The following sections describe how we implement the proposed model for designing availability, security and performance of the system with regard to the stages of Concrete Experience (CE), Reflective Observation (RO), Abstract Conceptualization (AC) and Active Experimentation (AE). Table II gives the summary of this implementation.

C. *Analyzing Designs for High Availability*

(CE) – Based on the initial configuration, the students are given a starting scenario to experience the potential loss of availability of the web application. For example, killing the software process of the web application or unplugging the network connection between the proxy server and the web application. The students can and should discover many other scenarios that can cause the loss of availability of the web application. For example, any single point of failure in the architecture design can cause the loss of availability such as down of the proxy server or web application due to coding bugs or device issues. The risk scenario log is updated at the end of this stage.

(RO) – At this stage, the students reflect on the symptoms and single points of failures affecting the availability of the system. For example, how did the proxy server handle the exception? Did the proxy server retries after failure? The

Table II. COURSE CONDUCT OF SOFTWARE QUALITIES SESSION BASED ON PROPOSED MODEL

Course Conduct	Concrete Experience	Reflective Observation	Abstract Conceptualization	Active Experimentation
Activity				
Analyzing Designs for Availability	Experience the potential loss of availability of the web application.	Reflect on the single points of failures for the system	Demonstrate solution using sequence and deployment diagrams	Implement the redundancy and clustering designs
Analyzing Designs for Security	Experience the potential security compromise of the web application	Reflect on the vulnerabilities and potential threats of the system	Demonstrate solution using sequence, network diagrams and firewall rules table	Implement the security countermeasures to mitigate the impact
Analyzing Designs for Performance	Experience the performance of the web application	Reflect on the performance results and potential bottlenecks of the system	Demonstrate solution using sequence diagram or table calculations	Implement the performance tuning designs
Design				
Key Learning Outcome	Identify the risks through concrete experiences	Analyze the root cause of the risk	Design the solution to mitigate the risk	Implement the solution and verify that the risk is mitigated
Duration	1 hour	45mins	45mins	2 hours
Conduct	Within class time	Within class time	Within class time	Within and beyond class time

students can discuss within their group and the instructor to understand the symptoms and identify potential root causes to the identified risks. At the end of this stage, the students have to prioritize the risks and identify the key risks to be addressed for the rest of the stages.

(AC) – The students conceptualize what can be done differently to mitigate the key risk. For example, how to design for redundancy and clustering of the web application to mitigate this risk? There can be many mitigation measures and the students have to discuss and draw sequence and deployment diagrams to illustrate their designs. Possible design considerations include horizontal or vertical scaling of the web application, how to check for failures and the failover mechanism of the web application instances.

(AE) – With the given network setup and source codes of the proxy and web application, the students can implement their idea to improve the availability quality of the system. For example, the students can implement and test a redundant and clustered web application configuration. The proxy server can be programmed to detect failure and failover to the redundant web application instance.

This cycle can iterate to address other risks in the risk log in terms of loss of availability. There is no 100% always available architecture and the students are encouraged to identify and address as many risks as they can.

D. Analyzing Designs for Security

(CE) – Based on the initial configuration, the students are given a starting scenario to experience the potential security compromise of the web application. For example, sniffing using OWASP Zap [13] expose the login credentials which are contained within unencrypted network traffic.

(RO) – At this stage, the students reflect on the potential threats and vulnerabilities of the system. For example, what are the root causes of the proxy server or the web application leading to this compromise? What are the other threats and vulnerabilities with respect to this two-tier web application setup?

(AC) – The students can conceptualize what can be done differently. For example, how to design for security such that the data are secure not just in transit but also in storage and archive. They can draw sequence, network diagrams and design firewall rules to illustrate their designs.

(AE) – With the given network setup and source codes of the proxy and web application, the students can implement their idea to improve the security quality of the system. For example, the students can create the digital certificate and implement secure socket layer (SSL) for the web application.

This cycle can iterates with other potential situations in terms of security compromise. The attack surface of the system comprises of potential vulnerabilities or weakness being exploited by threats in many areas from the application, devices and network. In this case, the application design can also introduce many other vulnerabilities such as SQL injection, clickjacking attacks and many more.

E. Analyzing Designs for Performance

(CE) – Based on the initial configuration, the students are given a starting scenario to first experience the initial performance of the web application. For example, a performance test using Apache JMeter [14] for a given set of concurrent users can help to create a baseline of the web performance.

(RO) – At this stage, the students reflect on their performance trends and identify potential bottlenecks of the system. For example, what are the levels of the CPU, memory and network utilization on devices hosting the proxy server and web application? Where is the bottleneck?

(AC) – The students can conceptualize what can be done differently. For example, how can they tune to improve performance by addressing the bottleneck identified earlier? Will implement a cache helps? They can illustrate their design using sequence diagrams or table calculations.

(AE) – With the given network setup and source codes of the proxy and web application, the students can implement their idea to improve the performance quality of the system. For example, if there is the bottleneck but the memory is underutilized at 40%, can the memory allocated to the web application be increased to improve performance?

This cycle can iterates with other potential situations in terms of performance issues. For example, if the memory bottleneck is resolved, another iteration might expose the bottleneck to be either the CPU of the device hosting either proxy server or web application or the network setup. In this case, can the maximum connections to the proxy server and the web application be tuned to improve performance?

VII. Course Feedbacks And Analysis

A. Course feedback

The students are exposed to the two teaching methods in two sessions of the course: Session 1 is conducted in a lecture format. Session 2 is conducted based on the proposed model. We wish to seek feedback using a survey from the same group of students on the effectiveness of the proposed model in session 2 as compared to session 1.

B. Course Analysis

The survey comprises of metric-based questions rated on a 5-point Likert scale metric ranging from 1 – Strongly Disagree, 2 – Disagree, 3 – Neutral, 4 – Agree and 5 – Strongly Agree. The question being asked for session 1 is "Degree to which the lesson and activities help in your understanding of the architectural concepts?" We posed the same question for session 2 in terms of each stage (CE, RO, AC and AE) of the proposed model. The last part of the survey is for the students to provide comments about the session 2 using the proposed model.

Fig. 2 shows the distribution of the responses to the question for each of the teaching method and stages. The survey indicates a consistent trend from the students to rate higher for the stages of the proposed model in session 2 over session 1 which is based on traditional lecture-based format. The weighted average in Table III summarize the weighted average of session 1 and 2 lessons with a similar trend. The highest rating of 4.51 goes to the abstract conceptualization stage when the students start drawing their solutions conceptually using diagrams. The ability to learn from others and compare different solutions to solve a concrete problem at this stage might be the reason behind the higher ratings.

We are also interested in their comments about the revamped course and perform sentiment analysis on their comments using four cloud services – (1) Azure Machine Learning For Sentiment Analysis, (2) Amazon Comprehend, (3) Google Cloud Machine Learning Analyzing Sentiment and (4) Aylien Sentiment Analysis. Azure Machine Learning For Sentiment Analysis [15] is a cloud-based service provided on the Microsoft Azure platform with both API and add-on for Excel, allowing batch analysis of text to be done quickly. Amazon Comprehend [16] is also a cloud-based service provided on the Amazon Web Services platform with API service available. Google Cloud Machine Learning Analyzing Sentiment [17] is an API service provided by Google Platform as a Service and Aylien Sentiment Analysis [18] is another online cloud-based API service provided by Aylien. All four services are free within a certain degree of usage. We first pre-processed the data before using the cloud services. For example, removing of comments such as "NIL" or "None".

Using four cloud services provide us opportunities to validate the results but also introduce challenges due to their designs. The type of sentiments returned can differ and calculation of the sentiment scores by all four services are not the same as shown in Table IV. The sentiments returned for both services (1) and (4) are either a positive, neutral or negative sentiment. For services (2) and (3), the sentiments can be positive, neutral, mixed and negative. A mixed sentiment contains text indicating both positive and negative sentiments and usually, the text comprises of more than one sentences. The score returned for service (1) is between 0 (negative) to 1 (positive) for the key sentiment and the higher the value, the more positive is the sentiment and vice versa. However for service (2), multiple scores are returned with a value between 0 and 1 indicating the level of confidence of each sentiment with the all the four scores adding up to 1. We only use the sentiment with the highest score as the key sentiment in our analysis. Service (3) returns a score between -1.0 (negative) to 1.0 (positive) with respect to the overall emotional leaning of the text. Service (4) returns a score between 0 and 1 for the key sentiment only.

For analysis, we seek to find common ground among the four services and decide to evaluate the percentage of the positive, neutral and negative sentiments of each service followed by the scores for significant negative or positive comments. Table IV shows the results of the sentiment analysis. All four services indicate a higher trend of positive comments which is encouraging. However, there are some differences between the sentiments results returned from the four services. Interesting, there are comments analyzed as negative sentiments by services (1) and (4) but are categorized as neutral or even positive by services (2) and (3). For example, this comment is analyzed as negative sentiment by services (1) and (4) but positive by services (2) and (3) "I have a feeling that there's a lot more concepts which can be covered and learned under architecture design and analysis." Another example is the comment "The security part of architectural concepts is interesting but in my opinion, it seems to be more in line with the module, Information Security & Trust.". This comment is analyzed as negative sentiment by services (1) and (4) but neutral sentiment by services (2) and (3). However,

TABLE III. WEIGHTED AVERAGE OF THE SUMMARY RESULTS

Teaching Method and Stage	Weighted Mean / Median / Mode 1 – Strongly Disagree, 2 – Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly Agree
Session 1	
Lecture Format, Quizzes, Discussion	4.17 / 4 / 4
Session 2	
Concrete Experience	4.30 / 4 / 5
Reflective Observation	4.44 / 5 / 5
Abstract Conceptualization	4.51 / 5 / 5
Active Experimentation	4.25 /4 / 5

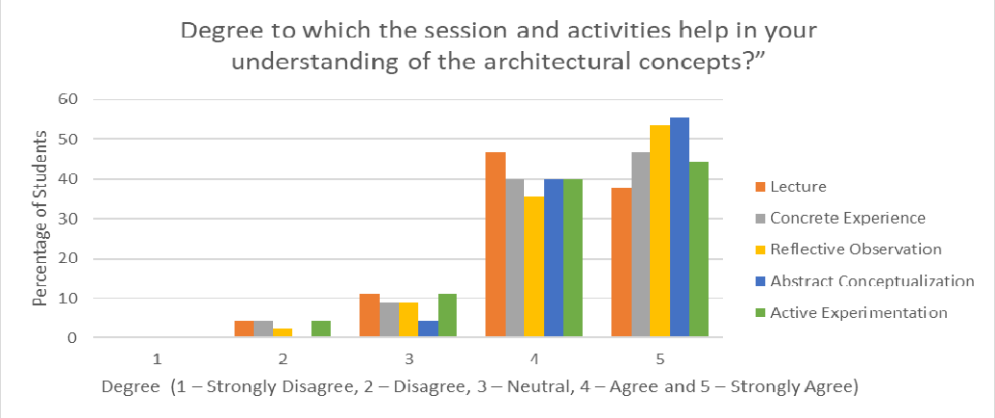

FIG. 2. SUMMARY OF THE SURVEY RESULTS

Table IV. SENTIMENT ANALYSIS OF THE COURSE COMMENTS

No	Sentiment Analysis Service	Types of Sentiments	Calculation of the Score	Sentiment			
				Positive	Neutral	Mixed	Negative
(1)	Azure Machine Learning Web Server	Positive, Neutral, Negative	0 (negative) to 1 (positive)	**75.00%**	13.89%	N.A	11.11%
(2)	Amazon Comprehend	Positive, Neutral, Mixed, Negative	Degree (0-1) of confidence for each sentiment	**66.67%**	27.78%	0%	5.56%
(3)	Google ML Analyze Sentiment Service	Positive, Neutral, Mixed, Negative	-1 (negative) to 1 (positive)	**83.33%**	16.67%	0%	0%
(4)	Aylien Web Service	Positive, Neutral, Negative	Degree (0-1) of confidence for the key sentiment	**86.11%**	2.78%	N.A.	11.11%

such cases are not typical, accounting for less than 2% of the comments.

The comment with the highest significant score among negative sentiments is related to the duration of the lesson – "I feel the time given is too short to cover all relevant concepts such as performance and security part." It has a sentiment score of 0.18 by service (1), 0.44 for negative sentiment by service (2), 0 for service (3) and 0.68 for negative sentiment by service (4). This can be due to this revised course design as conducting the session 2 based on the proposed experiential risk learning model requires more class time. We will need to review and potentially reduce the course contents to address this concern. On the other end, the comment with the highest significant score among positive sentiments is – "I really enjoyed your AA lessons and would strongly recommend anyone to take your course!" It has a sentiment score of 0.97 by service (1), 0.99 for positive sentiment by service (2), 0.9 by service (3) and 0.99 for positive sentiment by service (4).

VIII. GENERALIZATION OF THE MODEL

In the earlier sections, we describe how we design the software architecture course based on the proposed experiential learning model. However, the applicability of this model is not confined to the domain of software architecture model and can also be applied to other domains of software engineering or other subjects with a strong relationship with risk management. In this section, we generalized the model to apply to two other domains of the software engineering - gathering requirements and software testing. The key ingredients to apply this model is the identification of significant risks applicable to the domain, the ease to demonstrate these risks and verify the effectiveness of the controls. This proposed model effectively make risks a first-class citizen in the design of the lesson.

A typical sequence of topics in a requirement management lesson is to teach the importance of requirements gathering activity, the pitfalls to avoid, the method to gather requirements and lastly practice this method with a scenario. If the proposed model of learning is applied for this activity, a list of gathered requirements for the same scenario, the design deliverables and an executable product are given to the learner upfront. The product does not meet the user's expectations and the learner have to assess the requirements gap, the impact of these gaps and what is the cause or flaws in the requirements gathering that lead to this gap. With these understandings, the learner then can design a countermeasure such that the risk can be controlled. The learner can review the existing methods used to gather requirements and revise the steps to ensure the gaps in requirements can be captured. The last step is to implement this revised method and verified that the requirement gap is closed, the design deliverables are updated and the product is developed to the user's expectations. In this example instead of the learner being taught the method to gather requirements upfront and practice it, the learner experiences the impact if the method is not practiced correctly and subsequently attempt to correct it. During this process, the learner not only understands how the method works but also the impact of wrong requirements while fixing the requirements gap.

In software testing, test designers typically create the test plan and cases based on the requirements and design deliverables. During testing, the tester will test the actual product following the test plan and test cases. If the proposed model of learning is applied for this activity, the test plan and a list of test cases, the design deliverables and an executable product are given to the learner upfront. However, the product contains flaws that are not within the range of test cases. The learner has to assess the impact of these gaps and what is the cause or flaws in the generation of test cases that lead to this gap. The last step is to implement the missing test case, verified that the gap is closed, the design deliverables are updated and the product is developed to the user's expectations. In this example instead of the learner practicing the method to produce the test plan and test cases, the learner experiences the impact of the test plan or the test cases being implemented incorrectly and attempt to correct it. During this process, the learner not only understands how the method works but also experience the impact of incorrect or insufficient test cases to the final product.

We do note that in the above two examples using proposed model, there are additional effort to prepare not just the deliverables for a particular activity such as user requirements or testing cases, there is also a need to prepare the deliverables for the subsequent phases including the final product. However, this consequence also enables the learner to better experience the specific domain in a more concrete manner. This aspect is particularly useful when the domain contains abstract topics that are not easily comprehended by learners based on a classroom lecture format. Another potential problem during the course design using the proposed model is that the instructors are required to be able to articulate the lesson based on diverse real-world experiences. This is necessary for the design of the risk scenarios and appreciates the significant impact due to the risks. The similar issue would be having someone teach the risks of designing the wrong architecture of buildings who had not actually designed various kinds of buildings.

IX. THREATS TO VALIDITY

In this paper, the proposed experiential learning model is applied to the students taking the architectural analysis course in one semester. Although the course is conducted for four sections of students over the semester, we acknowledged that the study could be further validated for sections conducted over subsequent semesters.

The profiles of the students play a part in this study. We conduct this course for students taking an information systems program who are familiar with programming, design and business solutions. However, they are not exposed to the level of technicality as one in a computer science program.

The application of this model depends on the instructors' conduct of the class and interactions with the students. Although the class conduct for the four sections is based on the same instructors, there is still the potential threat that whether the study results will be valid for other instructors. We can only conclude that when we have other instructors teaching the same course using the proposed model.

X. CONCLUSION

In this paper, we showed how our proposed experiential risk learning model could be applied to the design of a software architecture course for undergraduates. Our findings show that the students preferred this model consistently when compared each stage of the model against the traditional lecture-based lesson. The sentiment analysis of their comments also shows a positive trend consistently across the four cloud services, giving us the encouragement to continue to validate this model. We also demonstrate the applicability of this model with the generalization of the model to other domains of software engineering. The applicability of the generalized model remains to be validated and we will seek opportunities to do that in future studies. We hope our study demonstrate to educators how to use the proposed experiential risk learning model as another teaching method in the course design for active learning and alleviates the challenge to design a software architecture course for undergraduates.

REFERENCES

[1] "ANSI/IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems," 2001. [Online]. Available: https://standards.ieee.org/findstds/standard/1471-2000.html.

[2] P. Kruchten, "The Software Architect," Springer, Boston, MA, 1999, pp. 565-583.

[3] E.L. Ouh and S. Jarzabek, “An Adaptability-Driven Model and Tool for Analysis of Service Profitability.” In: Nurcan S., Soffer P., Bajec M., Eder J. (eds) Advanced Information Systems Engineering. CAiSE 2016. Lecture Notes in Computer Science, vol 9694. Springer, Cham

[4] C. R. Rupakheti and S. V. Chenoweth, "Teaching software architecture to undergraduate students: an experience report.," in IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE), 2015, 2015.

[5] M. Galster and S. Angelov, "What makes teaching software architecture difficult?," in IEEE/ACM International Conference on Software Engineering Companion (ICSE-C), 2016.

[6] D. A. Kolb, “Experiential learning: Experience as the source of learning and development. FT press, 2014.

[7] G. G. Mitchell and J. D. Delaney “An assessment strategy to determine learning outcomes in a software engineering problem-based learning course. International Journal of Engineering Education”, 20(3), 494-502, (2004).

[8] B. W. Boehm, Software risk management: principles and practices. IEEE software, 8(1), 32-41, 1991.

[9] J, Liddle, (1998). Risk management: Walking the tightrope. The Journal of Experiential Education, 21(2), 61.

[10] K. L. Minh, E. L. Ouh, and S. Jarzabek. "Mood self-assessment on smartphones." In Proceedings of the conference on Wireless Health, p. 19. ACM, 2015.

[11] D. Koznov, D. Luciv, H. A. Basit, E. L. Ouh, and M. Smirnov. "Clone detection in reuse of software technical documentation." In International Andrei Ershov Memorial Conference on Perspectives of System Informatics, pp. 170-185. Springer, Cham, 2015.

[12] "ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.," 2011. [Online]. Available: https://www.iso.org/standard/35733.html

[13] OWASP Zed Attack Proxy Project [Online]. Available: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[14] Apache JMeter [Online]. Available: https://jmeter.apache.org/

[15] Azure Text Analytics API - Detect sentiment, key phrases and language from your text [Online]. Available: https://azure.microsoft.com/en-in/services/cognitive-services/text-analytics/

[16] Amazon Comprehend - Discover insights and relationships in text [Online]. Available: https://aws.amazon.com/comprehend/

[17] Google Natural Language - Analyzing Sentiment [Online]. Available: https://cloud.google.com/natural-language/

[18] Aylien Text Analysis API - Natural Language Processing for Effective Understanding of Human-Generated Text [Online]. Available: https://aylien.com/text-api/